CHAPTER 6

TECHNOLOGICAL IMPLICATIONS ON SOFTWARE DEVELOPMENT

Porf. Dr. Mohamed M. El Hadi

ABSTRACT

The World - wide computer industry has been characterized since its inception by continuous innovation, improvement and rapid change. The developments have resulted in the growing importance of software, i. e. the programs that run computers and allow them to communicate with each other through data network.

Egypt as a newly developed country is trying to seek a share in the on-going global information of its software industry through a variety of policy and industrial measures. At the same time, a growing number of software development firms and organizations are beginning to develop software to be marketed domestically and abroad.

The world - wide software industry is still very much new mental industry. Any assessment of international software industry and market has to rely on insufficient, inconsistent and unreliable data. This makes the projections for the future software products and services an exceedingly risky undertaking. Nevertheless, such projections have been made and the range from 70 - 180 billion US $ by 1990 which has doubled by now.

There are at least four major forces which promoting rapid changes of software industry in the world at large, and in the Egyptian situation in particular; These forces are as follows:

(1) A productivity bottleneck in programming, with software and software - related support activities now accounting for the overwhelming of total system costs.

(2) The move away from single vendor solutions : as the typical way for organizations to meet their information systems needs towards customized, integrated, multi - vendor hardware and software solutions.

(3) The increasing emphasis on software production and sales by hardware vendors, leading construction by large and medium sized firms.

(4) An expansion and fragmentation of industry resulting in large number of independent software vendors.

The technology of software development as related to tools and methods, as defined broadly as procedural methods, organizational models, and technological knowledge of development tools which are used to transform inputs into outputs, is becoming an increasingly important element in international competitiveness of organizations and management of software production process of firms.

This paper reviews some new software development practices being introduced by software development firms to assist in software engineering and software control costs and improve the quality of both the final products and the process of developing software. Some new technologies that will affect the entire software production process and that will require a reorientation in thinking about the investment and technological options are examined in this paper.

Therefore, the paper highlights the Capability Maturity Model (CMM) which gives software development firms and organizations a framework for evaluating their development processes and taking appropriate steps to improve them. The model's five levels include the initial level at which business adopt the program. In the second level, software development firms are able to deliver their software products on a consistent schedule and with a specific quality level. In the defined level, the developers begin to understand what practices let them to deliver consistently, and they disseminate the practices across their firms. At the management level, the practices are quantified, and at the optimized level, the new programs are established as routines and maintained with constant improvments.

INTRODUCTION

Software development firms and organizations in Egypt as in other developing countries are still struggling with inappropriate tools and methods utilized in software development process. This problem is due to the following factors :

(1) Lack of knowledge on the discipline.

(2) Lack of experience.

(3) Difficulty of measuring the development effort accurately.

(4) Implementation of designs with which poor quality does not surface until the finished products is either tested or installed for operation.

(5) High life cycle costs resulting from a system that was not designed for reusability or maintainability.

Without establishing processes, methods and techniques, software development firms and organizations often solve the same problems over and over again. The average large software project continuous to cost twice as much as its initial budget and it is completed a year or two behind its schedule; approximately 25% of such projects are never completed and remaining about 75% require inordinate amount of maintenance.

Therefore, the adaptation of new technologies and the more efficient computer programs are crucial for software industry. It is claimed that the market directs the evolution of software industry. The new market dominated nowadays, by the ***Internet*** will be important in short - term, which Egypt has to consider in its efforts to develop its software industrial sector.

The utilization of new technologies in software development and engineering is considered the basic guideline to have a share in the interantional market of software products. Therefore, a strategy of software industry should be underlined and emphasized by the concerned parties. There are two advantages of such a strategy:

* the first one, should bring the software development firms closer to customers or users, who, all over the world are not looking for one time transaction. They want at least a ten year partnership. This model helps to understand the challenges that firms and organizations are going through.

* The second software strategy, deals with by breaking up into industry units. we can maintain our entrepreneurial culture and continue to grow like a small company or firm. Therefore, a core software development center should be maintained for the firm's human resources and financial module. but responsibility for all sales and services will rest with the individual business units.

Taking, these two strategies in consideration, software products and new technologies should be patented for preserving intellectual property rights. For example, the U. S. Patent and Trademark Office (PTO) issued about five thousands patents for software products in 1993 [1]. This is an eightfold increase over the previous decade. The number was expected to exceed 5500 with the year 1995. It is stated that the most popular areas of software patents are; image processing, followed by network and communication systems. Anecdotal evidence suggests that U. S. and Japan hold half of all the software patents issued world wide.

SOFTWARE DEVELOPMENT PROCESS

A major promise in software development process is that the quality of a piece of software which is governed largely by the quality of the process used to develop and maintain it.

Continuous support for a software development improvement effort requires at least tow things : a clearly defined improvement model to follow, and success at applying the model in the firm or organization.

Manufacturing enterprises are always looking for more efficient ways of producing products because they realize that an efficient process yields lower costs, better quality and increase customer satisfaction. Software developers and producers are no different from their hardware counterparts in that they want to use the best software development process available.

In any mature software process, the methods, techniques and technology are to be used effectively and produce reasonably consistent results. Improvements in quality and productivity occur in part from automation. But in an immature software developed process, unpredictable results occur : formal procedures, cost estimates and project plans are lacking; and technology is used on ad hoc basis;. The prospects of software development in Egypt will depend largely and increasingly on the processes carried out in the software development life cycle to be followed. There is considerable middle ground between these two development processes extremes. The Capability Maturity Model (CMM) for software developed by the Software Engineering Institute at Carnegie - Mellon University is a process model that provides excellent guidance to improve software development processes.

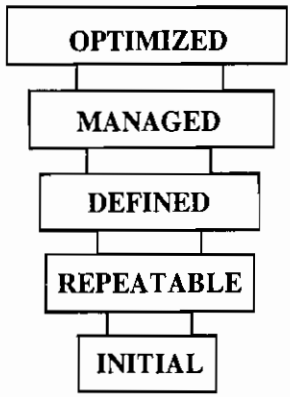
I. THE MATURITY MODEL :

The Capability Maturity Model (CMM) [2] is used to evaluate and improve the way software is built and mentained. This model was first released in 1987, it was originally based on the experience of members of the Software Engineering Institute. The "CMM" has been continuously improved and refined since 1987 through successive revisions based on industry - wide and world - wide input. Yet, even though it is based on experience, it is only a model, which is an abstract, general framework describing the process used to develop and maintain software.

Like any model, it requires interpretation to be used in specific context. The approach used by "CMM" is to describe the principles and leave their implementation up to the managers and technical staff of each firm which will tailor "CMM" according to the culture and experiences of its own environment.

Perhaps the most well - known aspect of the CMM is its description of fife stages or maturity levels of a firm's software development process. These levels are shown in the following figure.

Figure 1: Software Development Maturity Model



This model is utilized if firms and organizations interested in software development can judge the effectiveness of their software process has five maturity levels : initial, repeatable defined, managed and optimized.

(1) The Initial Process Level :

A software development firm has ill-defined procedures and controls. It typically, operates without project controls and does not integrate tools and techniques with the process. Coding and testing are being dominant activities. Established procedures, if they exist, are usually abandoned in a crisis. Therefore, This level is described as ad hoc, poorly controlled, and often with unpredictable results in terms of schedule, effort, and quality.

(2) The Repeatable Process Level :

A Software development firm or organization established basic project controls, such as : project scheduling, coding standards, software quality assurance, and change control. Mechanisms are in place for ensuring that the designer or the design team understands each software requirement. Statistics may be gathered on software code and test errors. The strength of the software development firm may stem from its prior experience at doing similar work, but it may face major risks when presented with new techniques. Therefore, in this level, the outputs of the process are consistent (in terms of schedule, effort and quality) and basic controls are in place, but processes that produce these results are most defined and understood.

(3) The Defined Process Level :

Standards and methods for technical and management activities required for software development are established at this level. These specifically include de-

sign and code reviews, training programs, and increased organizational focus on software engineering, including measuring specific tasks in the development process. Some uncertainties remain about the value of the measurements, the best ones to use and the appropriate response to reviews. In this level, the software engineering practices that lead to consistent output are known and understood, and are used across the whole firm or organization.

(4) The Managed Process Level :

A firm typically has a minimum set of process management for each level in the software development life cycle, and it conducts extensive analysis of the data gathered during reviews and tests. Automated tools and techniques are used increasingly to control and manage the development process, as well as to support data collection and analysis. Therefore, within tthis level, the defined elements from the proceding level No. 3 are quantitatively instrumented, so that the level No. 5, the optimization level can be achieved.

(5) The Optimized Process Level :

This level exists in firms and organizations in which their development process operates smoothly as a matter of routine and continuous process improvement is conducted on the defined and quantified processes established in the previous levels. Therefore, these firms at this tevel have achieved a high degree of control over their process, have automated data gathering and typically have a method for improving and optimizing these operations. This includes identifying and replacing obsolete technologies, more sophisticated analysis of error and test data, and the introduction of error cause analysis and preventive studies [3].

The following figure (2) illsutrates these five levels of the "CMM". The items listed are called key process areas. These areas determine an organization's software development maturity.

Figure 2: The fife levels and key process areas
of the capability maturity model

Level 5
Optimizing (Continually Improving Process)
* Process Change Management
  # Technology Change Management
  # Defect Prevention

Level 4
Managed (Predictable Process)
* Software Quality Management
* Quantitative Process Management

Level 3
Defined (Standards, Constituent Process)
  # Peer Reviews
  # Software Product Engineering
* Inter group Coordination
* Integrated Software Management
* Organization Process Defintion
* Organization Process Focus

Level 2
Repeatable (Discipline Process)
  # Software Configuration Management
  # Software Quality Assurance
* Software Subcontract Management
* Software Project Tracking and oversight
* Requirements Management

Level 1
Initial (Ad Hoc, Chaotic)
* Managerial Processes
  # Technical Processes

Thus, "CMM" is seen as a maturity or growth model in which a firm works its way up the five levels and, even after having attained level 5, is still in the process of continually improving and maturing.

Each of the five levels is also defined by the key processes associated with it. There are 18 key process areas that make up the five levels (see fig. 2: above). These processes are to be chosen because of their effectiveness in improving an organization's software process capability. They are considered to be requirements for achieving a maturity model.

Managerial processes are those that primarily affect the way operates to make decisions and control the project.

Technical processes are those tools that primarily affect the way developers operate to perform the technical software work.

Some software development firms and organizations. particularly in the U.S. started to use the capability maturity model to assess their software development process and software management. The importance of such assessment is to indicate the maturity and tecnological levels at which a firm is operating. More importantly, it may indicate the strong and weak areas of the firm's software development capabilities, thus identifying immediate imporvement priorities, interim improvement goals and progress measures.

One implication of the maturity process model for the software procedures in Egypt, for example is clear i.e., technology and managing the process of software engineering tangible products will become increasingly impotant. An understanding of the maturity model and how it might be of practical benefit to software development firms and organizations in Egypt, should be a research priority to be conducted.

II. GOALS OF THE CMM's KEY PROCESS AREAS :

The CMM provides a structure for each of the key process areas. Also, each key process area has one more than goals that are considered important for enhancing process capability.

The goals for the three of the key process areas in level 2, are shown in the following:

Level 2 : Repeatable. basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier success on projects with similar applications.

*** Requirement Management :**

1 - Requirements are controlled to establish a baseline for software engineering and management.

2 - Plans, products, and activities are kept consistent with the requirements.

*** Software Project Planning :**

1 - Estimates are documented for use in planning and tracking.

2 - Project activities and commitments are planned and documented.

3 - Affected groups and individuals agree to their commitments related to the project.

*** Software Project Tracking :**

1 - Actual results and performance are tracked against the software plans.

2 - Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the plans.

3 - Changes to software commitments are agreed to by the affected groups and individuals.

Finally each key area has five common features or attributes :

1 - commitment to perform and describe the actions needed to ensure that the process is established and will endure and typically involves policies and senior management sponsorship.

2 - Ability to perform and describe the preconditions that must exist in the project or organization to implement software process competently. Ability to perform typically involves resources, organizational sttructures, and training.

3 - Activities to performe and describe the roles and procedures necessary to implement a key process area. These typically involve established plans and procedures performing the work, tracking it, and taking corrective actions as necessary.

4 - Measurement and analysis which describe the need to measure the process and analyze the measurements.

5 - Verifying implementation describes the steps needed to ensure that the activities are performed in compliance with the process that has been established. Verification typically encompasses reviews and audits by management and software quality assurance.

The intent of CMM is to describe needs to be done to develop and maintain software reliable as well, not how to do it. CMM further describes practices that con-

tribute to satisfying the intent of these attributes for each key process area. Any firm or organization can use alternative procatices to accomplish the goals of the capability maturity model.

III. THE CHALLENGES OF THE CMM :

It usually takes most software development firms about a considerable time (about 2 - 3 years) to go from level -1 to level - 2 of the capability maturity model's compliance. However, based upon sound business reasons, the general management of the firm could commit itself to reach level - 3 of this development model in less time (about a year and half only) and assign few people to do the needed tasks.

During the planning stage, it could be discovered that because this is a reasonable development process program, level - 2 could be completed in less time with less people than the previous practice, because the CMM provides immediate benefits to the firm which applies it.

Therefore, the software product development team could reach level - 2 of the CMM compliance within a few time of staring the project, beginning with investigating the project and continuing implementing it. Throughout this period, two deployment cycles with the development team could be completed : the internal audits of the team processes verify what is being operated at level - 2 of the CMM, and several more audits of all the development team are executed externally.

SOFTWARE DEVELOPMENT LIFE CYCLE

I. INVESTIGATION PHASE :

To investigating the current practice, a technique called **Software Process Profile** could be used. This technique which was developed by Hewlett-Packard and the Software Engineering Institute at Carnegie - Mellon University, is considered an assessment of the state of an organization's software development process which identifies strengths and weaknesses, highlights the process improvements which the organization values most, and recommends areas for change. The profile uses the capability maturity model as a standard for evaluating the software process. Through the use of questionnaires and open-ended interviews, the profile could provide results for the key process areas of the software development maturity model.

To complete the profile, developers and managers of the project and the firm must fill in a questionnaire that is to evaluate the firm's software process maturity against the capability maturity model's requirements. The results of this questionnaire are to be compiled into a software process profile for the firm. An example of typical survey questions to assess the software project tracking and oversights are :

* Are the project's planning activities and deliverables tracked, (e. g., schedule, effort and tests)? Fully or Partial.

* Are the actual results compared to the estimates throughout the project? Fully or Partial.

* If there is a variance, does someone take corrective action or explain why variance has occurred? Fully or Partial.

* Are changes to the activities, deliverables and schedule discussed with the people who will be affected? Fully or Partial.

* Does someone review the project results regularly with development team? Fully or Partial : or Almost Always, or Often or Seldom.

Therefore, in applying the capability maturity model effectively, one of the critical steps in getting started is to understand the model in detail. The specifications of the model are to be reviewed, the interpretation should be developed. and the model itself should be translated into a language to be applied by the firm. This is considered an important step because the development maturity model contains many practices that are better suited to large firms, and it is necessary to interpret which of these practices would apply to the firm. Also ideas of how best to deploy these practices should be developed.

Several process assets of the development firm could be leveraged to support the key process areas requirements. These assets are software live cycle; formal documentation for patches, releases and defect management procedures; and several informal methods of documentation for specifications, design and testing.

A need for a ***process architecture*** is to be needed when it is attempted to describe what deliverables would be needed to support the definition of processes. A process architecture is analogous to other types of professions such as engineering architecture (e. g., buildings, software, hardware, etc.).

Therefore, a process architecture describes the layout of computers needed to build a system for a specific purpose.

The elements of a process architecture are as follows :

Policy * Specifies what will happen,

* Sets the cultural expectations,

* "That's how the firm does things".

Procedure * Specifies how it will happen,

* A set of steps for doing something.

* May specify who does what when.

Checklist * Specifies what or how in abbreviated format,

* A short form of procedures for easy reference or verification of actions.

Template * Specifies the content or quality of what will happen,

* Provides guidelines for creating necessary work products.

Training * Provides organized information on processes that individuals need to perform their jobs,

* Covers policies, procedures, checklists, tepmplates, or instructions,

* May be formal classroom or informal orientation.

Work Product * Specifies a plan or results,

* Created as an output of applying a defined process,

* May need to be "managed and controlled".

II. FORMAL PROJECT PLANNING AND DECISIONS PHASE :

To give this improvement project the greatest chance of success and reduce the overall risk for the project, a formal project plan is to be developed covering every phase of the defined work and the timing for deployment of the processes. This plan is to review and to be approved by the team's members before beginning the implementation.

Several key decisions need to be made about how the project deliverables would be designed, reviewed, approved and deployed into the software development operations. Also, several models for the process creation, approval and deployment are to be examined and discussed before it is decided to use a definedeploy approach to be mapped into each development life cycle phase (i.e., requirements, design, implementation and test). This could provide a structure within which process deliverables could be refined and improved.

The software development process needs to show results as early as possible to capture the attention of the organization, and to keep things focused on process improvement. To accomplish the objectives of the early showing results, the deployment stages of the development process are to be designed. This tactic provides visible results and feedback to the organization by coinciding with the life cycle phases.

A series of short steps for defining, reviewing and approving the policies, procedures and templates to be followed are needed before deploying them to the soft-

ware development teams. Communicating what is expected, describing changes from the way it is used to do things before, and providing group of individual training in new methods are considered very important steps in deploying the process changes. It is known, that the development project teams need to understand the rules early, so that the project could proceed smoothly.

During the ***assessment and planning*** of the improvement development process project, it is recognized that many practices could be used at level - 2 of the CMM compliance. Therefore, it is important to leverage as many of the current practices and procedures to minimize the effort required and reduce the amount of change being introduced. For example, standard software life cycle development toolset, good practices and tools in configuration management, defect management, inspections, coding and testing are among the practices to be used. Also, a customer satisfaction survey and software metrics are of much help.

The development environment consisted of a suite of tools integrated together with the main development software tool such as CASE, ITE or SoftBench of HP are much needed as a rapid prototyping tool. this tool is to provide the framework of editing, compiling and debugging. Capabilities of these tools are also being used to assist in program understaning through all graphs and documentation of object oriented designs of the new features. Integrated, in such development tools, is a configuration management tool for managing source code, and the ability to access defect tracking. Having these tools is considered a major factor in maintaining developer productivity while making process changes in other areas.

Inspections and **software** metrics should be established as a part of the organizational culture. Although, both of these areas are considered elements of the level - 3 of the CMM, they are also, required for level - 2 to maintain the benefits which are achieved project with these practices. Because inspections and metrics (defect tracking, schedule slippage, and effort reporting) are institutionalized in the engineering and software management practices, it is recognized that this would be distinct advantage for the development of the software maturity process of level - 3.

Also, in the planning phase, it should minimize changes within the areas of software configuration management, i. e., software quality assurance, subcontract management.

Therefore, methods for measuring the progress of the software project development are very important to be established in this planning phase.

The only way to do this is through auditing the phased stages of the software development after each major checkpoint.

III. REQUIREMENTS PHASE :

During the requirements phase of the software development life cycle, the key process areas to work on are practices for requirements management and project planning. By using readily available customer survey data, structured processes, management review milestones, and training, it would be able to reduce the time allocated to perform this phase. It is important to gain a deeper understanding quickly of the new features demanded by the customers, and to translate this information into work steps that would be needed in the design phase afterwards.

One of the problems that the requirements phase is to face reducing the list of possible things to be accomplished to a few high impact requirements that are to be accomplished within the schedule. Thus, to collect requirements information, survey questionnaires based on a user - centered design methodology are to be created to facilitate rapid feedback from customers using telephone surveys for example. The process consultants designed standard templates for developers to describe the customer requirements.

One criterion for determining what new features to include is an estimation of resources and effort needed to design the new features. The decision making, process is very essential element for narrowing the scope of the project and finalizing the work commitments for the next phase of the software development.

A ***Software requirements checklist*** should be planned. A portions of this audit checklist for requirements phase which is used to assess the software development firm or organization life cycle against the level - 2 of the CMM model contains the following factors :

** Responsibilities are to be assigned for planning requirements phase activities.

** Staffing is sufficient for the planned activities.

** Adequate training and tools are provided for planning activities.

** Requirements phase activities are documented in a requirements phase plan.

** Estimates of schedule are prepared and included in the requirements phase schedule.

** A software life cycle is identified or documented in the requirements plan and design phase.

** Work products for control of the project during the requirements phase are to be identified in the project planning documents.

** Commitments are to be negotiated with the business team managers and other affected groups.

** External commitments are reviewed and approved by the business team managers.

** Responsibilities are assigned for developing and maintaining the design phase.

** Adequate training and tools are provided for planning the design phase.

** A design phase plan is prepared according to document procedures.

** Design phase activities are documents in the design phase plan.

** Estimates of size, effort and cost for design phase planning are developed.

** Requirements are traceable in the design plan.

** Issues related to design are reported and tracked.

** Design phase plans are to be updated to reflect changes in requirements.

IV. DESIGN PHASE :

During the design phase of the software life cycle, the key process areas are considered the practices in project tracking (i. e., managing and controlling work products, especially changes in requirements), and in project planning for this phase. The success in applying the software life cycle and the level - 2 of the CMM model processes to this phase of work is evident from the following two major accomplishments: first, the team should complete every aspect of the design phase including reviews and inspections. In the past, design specifications and design reviews were cut short because of schedule pressures; Second, the team should be able to make decisions early in the project about eliminating features that would be too costly to implement.

For example, unit testing capability is a feature to be considered for the software development, but it is eliminated during the requirements and design phases because of the staffing trade- offs. This action substantially reduces the risk of schedule slippage later on.

V. IMPLEMENTATION PHASE : MANAGING THE CHANGE :

Getting the software development firm to adopt the changes necessary for the CMM compliance is considered an important step in implementing the new processes. Defining the new policies and procedures and team providing training is necessary, but not sufficient. Changing the processes involves changing the culture of the firm, and this is where it is critical to get everyone thinking about the changes in a positive way. This aspect of change management is approached by deciding what would be tried to accomplish the following goals :

* Demonstrate success with a phased approach.

* Leverage existing processes and minimizing some areas of change.

* Make the contributions of everyone in the development team very visible.

Therefore, managing change requires good communication of change, acknowledgement of the progress to be made, and encouragement toward the optimum goal. After each software life cycle checkpoint, the quality assurance team performs an audit by interviewing the team members using checklists as mentioned in the requirements phase. The software quality assurance members are actually considered the process consultants. They bring experience and maturity to the audit interviewing, reporting and follow - up consulting.

These audits have several major benefits :

First, The project team should know ahead of time that a process they are using during a phase would be objectively evaluated by independent team or firm. This has the effect of evaluating the importance of using a defined process;

Second, the audit interviews could cover critical issues and risks for the software development's products that are almost always a result of deviating from the project's plan or processes.

For example, during the audit a problem with inadequate staffing for test planning activities could be identified. The project plan calls for completion of test plans for the context feature changes before the design complete checkpoint. Therefore, the issues and risks should be reviewed by the management team and decisions are to be made to take corrective actions.

It is believed that the circumstances and development environment for the firms and organizations are not unique. Many firms are trying various quality improvement endeavors for software development and finding it very difficult to make the changes necessary to be more rigorous and disciplined in their development practices. It is also, believed that many of the essential ingredients to make a software improvement program should reduce the time span of the development cycle with utmost efficiency and reliability.

Therefore, the capability maturity model (CMM) provides an excellent framework for defining software development process improvement for small software development firms. Achieving level - 2 of the CMM status has largely been a matter of establishing a direction with leadership from top management than instituting a credible program for improving the practices used by software projects in planning and managing the work according to the guidelines of tthe CMM.

ASSESSMENT OF SOFTWARE STANDARDS AND METRICS

1 . ASSESSMENT OF SOFTWARE STANDARDS :

Standards for software engineering will take a long time to develop. But with the rapid technological improvements and increasing competitiveness characteristics of the software industry, government agencies and other large purchasers of software are using new techniques to evaluate contractors' abilities to develop software packages according to modern software engineering tools and methods. Also, a great number of software firms and organizations are starting using new evaluation techniques to asses their own ability to create critical software projects and assess their overall competitiveness.

One methodology, developed by the Software Engineering Institute of Carnegie - University for the U.S. Air Force [4], examines a contractor's capabilities in :

* ***Organizational and resource management,*** including software engineering training and the adequacy of support facilities. This also includes, quality assurance, process management, configuration control, and quality and quantity of resources.

* ***Software engineering support and management***, which includes the scope and use of conventions, formats, procedures and documentation during the software development phases, software quality assurance and data management. It also, includes the depth and completeness of the process and how it is measured, managed and improved.

* ***Technologies and tools,*** a contractor may use in the software engineering process some computer software development tools to cross - reference between modules and to design debug code.

Since this methodology has become only one additional part of criteria for procurement, it is unlikely that it will create a radical change in contractors who are selected for software development projects. But the evaluations will probably lead major software firms to manage their software development process more closely.

An awareness of software producers of these evaluation methodologies is important to future development.

Raising some of the questions presented in various methodologies, [5] may improve the software process in software development firms and organizations, and to some degree enhance their competitiveness in international export markets. These methodologies are usually concerned with standards and practices of the software firms in the preceding areas of contractor's capabilities.

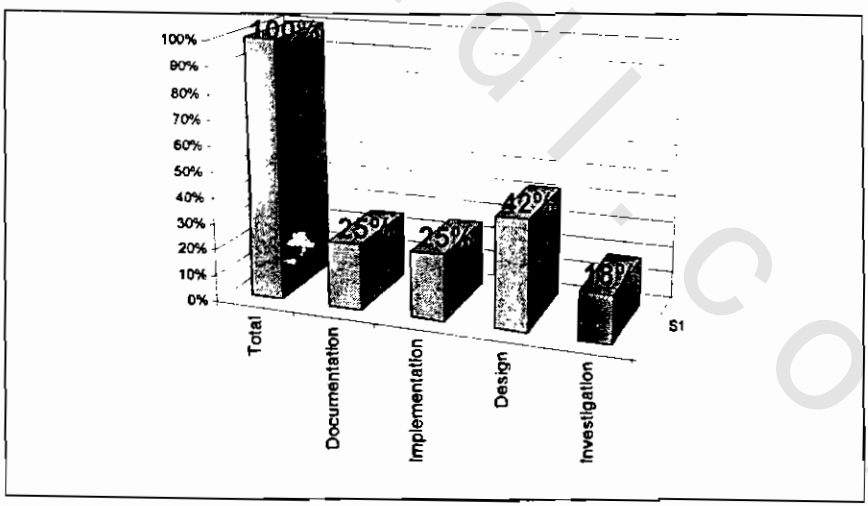
2 . ASSESSMENT OF SOFTWARE METRICS :

The status of software metrics, or measurement, is still somewhat immature and imprecisely defined, in the case of Egypt for example [6]. Nonetheless, measuring software products and software development can lead to substantial benefits for reasonable costs. Short - term productivity improvements, and the establishment of a common development environment, have been reported by companies that use software metrics. To many project managers in software firms, software metrics are a means for more accurate estimation of project milestones, and a useful mechanism for monitoring progress [7].

Basically, ***software metrics*** are a way of measuring the various attributes of the software development process. Attributes include the size of a program, its costs, the number of programming errors or defects, the level of difficulty of the project and the methods of communications required between members of the porject team.

The following figure (2) shows an example of using software metrics in software development in what is usually noticed as an ignored activity is documentation.

Figure 2 : Introduction of software defects



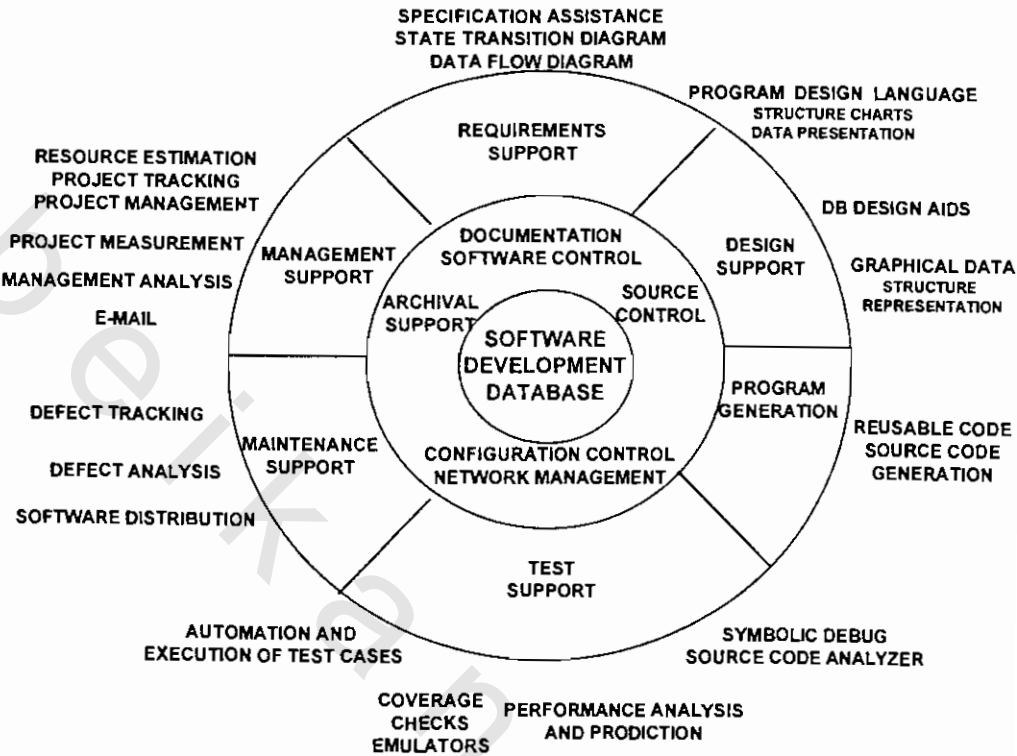
Documentation is frequently a low priority activity in software development process. Yet in the studies at IBM, the overall quality of software is very much a function of documentation. The above figure suggests that 25% of all software defects were in documentation delivered to customers [8].

Many tools are available for software metrics, some are in public domains to assist in collecting and examining measurable results during the various development stages; among of the already extensive utilized ones is the program called SOFTCOST which is used by the U.S. NASA's Computer Software Management and Information Center (COSMIC) which had developed and offered more than 1100 computer programs. The SOFTCOST package estimates software size, implementation productivity, recommended staff level, proper location, amount of computer resources required and cost of software documentation. Also, this program was designed to provide project managers with a comparison between their expectations of a project's development and industry - based statistical expectations of the project. The software metrics data provides an additional basis for budgeting time and effort to a project.

A complete integrated set of metrics' tools does not exist to be utilized by large software firms in their development processes in Egypt, up to now.

Anyway, the following figure (3) illustrates what Hewlett- Packard Co. considers an ideal software development environment which is reinforced by tools and metrics. The central database provides a common control mechanism for all important parts of a project [9].

Fig. (3) : Hewlett - Packard's software engineering productivity environment



Successful integration, collection and use of software metrics into software development process is the main objective of software firms and organizations using such tools. The time factor involved in learning tools and incorporating them into the entire development process has been the primary obstacle of time implementation. For example, the history of HP's Software Metrics Council shows that it took roughly three years of collecting and analyzing data using software metrics before there were sufficient data available to show measurable trends for the entire firm [10].

Software productivity is improving at a modest rate of about 4% a year compared with hardware, which is doubling in performance every four years [11]. To identify and promote best practice in software metrics, a project called ESPRIT brought together nine companies who have pioneered the use of metrics in Europe. The group of companies had significantly collective experience to offer. The results of this project were included in an interim report which describes an approach to metrics using current best practices.

Metrics will not take hold in a firm or company unless they are supported at the top of the organization. Senior management must understand the benefits and campaign on their behave, supporting middle management who can otherwise find themselves in an awkward position. But as this top level support, metrics need a sponsor group. It is recommended that an entity whether governmental or non-governmental should allocate full - time to tasks of promoting metrics and overseeing their collection, a status which is not existed also in Egypt up to now. We recognize that the importance of the existence of quality standards which provide the guidelines and criteria for software producers and can be quantified using metrics techniques.

THE IMPACT OF NEW TECHNOLOGIES ON SOFTWARE DEVELOPMENT

Research and development efforts in software development engineering have produced new methods that show promise for improving programmers productivity and software reliability. As the software development industry gradually becomes less labor - intensive over time, the quality and availability of skilled labor will become more important than labor costs. Pressures on the quality of software labor markets will increase competition among software firms and organizations, for a skilled labor in software production. But the introduction and adoption of these new technological tools and techniques is completely difficult and expensive, and they are thus adopted bit - by - bit as project budget allows [12].

1. AUTOMATED SOFTWARE DEVELOPMENT AIDS :

The conventional "Waterfall" software life cycle has been modified since its inception in the early 1970s. The major stages of software production (specification, design, coding, testing and maintenance) remain as planned, but efforts are underway to alter and automate many aspects of the software life cycle. Lower development and life cycle costs are being realized through better structured and documented software, program support library procedures, diagnostic aids, environmental simulators, test data management systems and applications generators. Such tools and techniques support different phases of life cycle. Some tools support early phases in the form of automated diagram drawing, screen painting, and error checking. Others give assistance by automating code generation and documentation.

Sizable applications programs can be generated using a very small number of user - language directives, which means that developing software with applications' generators can be a far more cost - effective pursuit than hiring programmers

to develop software one instruction at a time. Applications' generators are also valuable for their ability to develop quick prototypes of a desired capability.

The emergence of application' generators oriented around a database management system (DBMS) and report - generation capability has created an attractive approach for both software productivity gains and for software customization. Some software development firms in advanced countries which are using these tools have improved their productivity by factors of two as much as 20 across various phases of the software life cycle [13].

The facilities in three different tools : one for complex, real - time computer systems, the others for business applications are presented here.

These tools are reviewed briefly to illustrate the variety of ways in which software automation currently increases the productivity of software of all sizes and types.

(1) Auto - G Tool :

This development tool is produced by a small U.K. company, Advanced Systems Architecture (ASA). It is so far the only non U.S. system design tool that has been evaluated and supported by U.S. Strategic Defense Initiative (SDI). The ASA company has focused its operations on high reliable, high security real - time computer systems.

Auto - G tool uses a formal graphical notation, G, to enable software developers to build a system on a workstation in stages, from requirement specifications to code generation. When the design is complete, the Auto- G toolset provides a code generation that converts detailed design automaticatlly into a variety of programming languages, including for example, C and Ada languages. The Auto-G provides and includes the folowing :

*** Menu-driven, on-screen selection of design symbols,**

*** On-screen editing,**

*** Full graphical manipulation,**

*** Hard copy output low - cost graphic plotters, and**

*** Automatic checking of logical consistency of design.**

The excelerator of this tool has four basic facilities for autmating system analysis and design tasks :

(1) Automatic diagramming tool for drawing structured diagrams, such as Data Flow Diagrams (DFDs), structured charts, data models and control flow diagrams.

(2) Screen and report painters for prototyping user interfaces.

(3) Integrated dictionary for storing and cross - referencing all systems analysis and design information.

(4) Automatic checking and reporting the completeness and consistency of structured diagrams.

(2) The Cortext Application Factory :

This tool is used by systems analysts and programmers to develop and maintain medium - to - large information systems. It is, also, used to lead a developer through the steps of software development process and includes features similar to those of "Exceleratoes", including :

*** screen and report painters for prototyping user interfaces.**

*** A dictionary for storing documentation about a system.**

*** Automatic checking for completeness and consistency of program specifications.**

*** A code generation capable of automatically generating about 95% of the program code from program specification.**

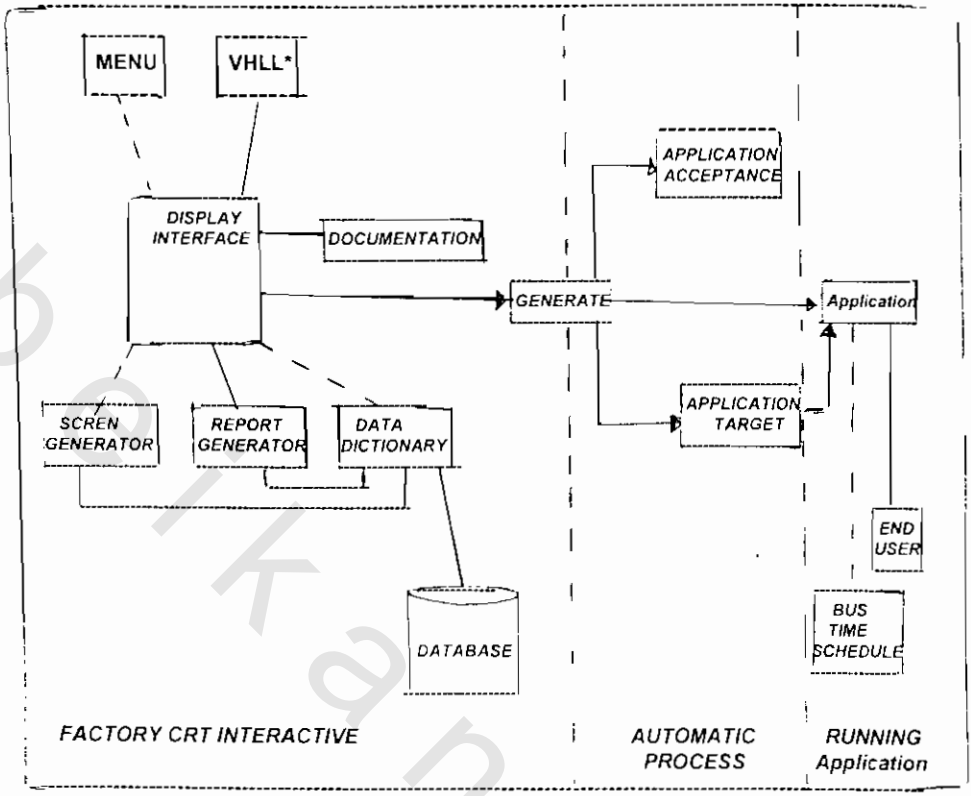
*** Automatic program documentation generator.**

Once accepted, the applications are translated into machine code by an optimizing compiler for fast computer run time and greater machine efficiency. Applications developed with the Cortex Applications Factory range form such applications as sales tracking, order entry, accounting, inventory and payroll to more sophisticated ones, such as regional price trends, tracking for commodities and orders for materials, etc. Also, applications range in size from a few files, screens and less than 50 database fields to those with several hundred files, screens and thousands of fields [14].

the developed applications generators can produce major benefits in productivity and case of implementation, use and maintenance if accompanied with good training and providing unexpected problems do not occur. Quantitative benefits include reduction in system life cycle costs and rapid development prototypes to ensure quick turn around for end user assessment. Among the possible quantitative benefits are the generation of documentation directly for specifications rather than program code, rapid iterative prototyping and greater control over maintenance.

Automated software documentation management techniques are gradually being introduced to software developers to improve quality of both software and documentation.

Fig. (4) : The Cortex Application Factory



* VHLL = Very High Level Language

Anyway, computer - aided software engineering, i. e. the CASE tools are available nowadays, for each of the life cycle development stages such as requirements analysis, detail design, coding and testing. However, these tools are not able, up to now, to replace or take over all the manual work in each of these stages. A reasonable approach is to combine these tools with the manual work, so as to maximize efficiency and improve the design quality.

(3) Integrated set of Tools :

Some tools now link hardware and software design (e. g., an integrated set of tools such as : C compiler, assembler, linker, simulator and debugger). These tools are developed by many companies such as Microsoft and Quantitative Technology Corp. (QTC), For example, the Q TC allows its software developers to evaluate design interactions of both hardware and software without waiting for a final version of either. A simulation informs hardware team how design changes will affect soft-

ware execution speed. Prototype is built, and software developers can see how code modification will work on the proposed hardware. Such tools are operating on mini- and mainframe computers [15].

Automated tools alone are not significant to ensure productive software environment. With no coherent methodology standards and set of controls, automated tools, sometimes, are just merely help developing a software project. Hence, they can also create an image of modern methods and high technology without substantively increasing productivity.

Successful use of automated tools requires a fit between the tool and the environment in which it will be used. The selection of the tools can be a complicated and confusing process. Tools must, in some ways, be similar to what a development company already does and knows; and they must be supported and maintained.

2. SOFTWARE REUSE :

If the programmer gets the software right for managing the movement of some thing from one place to another, it can be reused effectively regardless of what is being moved and how it is moving.

while reuse comes primarily through planning, design and organizational efforts, new technologies such as software repositories and components can encourage software reuse. Components which are defined as reuse software services bundled into an independent package, have been hailed recently.

Texas Instruments Inc., in Dallas, Texas is considered, for example, a leader in component reuse, identifies five types of components [16]:

(1) User interface controls, which are the technical widget, such as OLE controls,

(2) Technical infrastructure components, which include some bundled functionality, such as Internet connectivity,

(3) Domain - generic components, such as the coming authertification service which provides functionality across the establishment,

(4) Domain - specific components, which provide a complete package of business functionality for a partiular industry, such as telecommunications billing,

(5) Industry applications consisting of complete systems or templates for such functions as materials management or general ledgers,

It is relatively cheap and easy to get reusability for the first two types of components. Almost any developer can figure out how to use a package OLE custom control to create a sophisticated data window or provide some advanced functionality,such an image editor. The components do not require a lot of up - front planning, and they are readily available at low cost nowadays.

The best practice for software reuse involves the following characteristics :

* Build a strong architecture that provides the framework for reuse activities,
* Use a software - development process that promotes and controls reuse,
* Reuse more than just code, such as product and project templates, models, requirements, specifications,.. etc.,
* Practice domain development, such as identifying subdomains for a line of products, each with significant reuse potential. This helps reduce the scope of software development to a manageable size,
* Integrate reuse into project management, quality initatives, and software development. activities,
* Emphasize partnering to achieve reuse across product boundaries,
* Use automation to support reuse, including automated process improvement, software repositories, configuration management, domain development engineering, software factories, and integrated CASE.

3. RAPID APPLICATIONS DEVELOPMENT TOOLS :

In spite of there are many rapid applications development tools, only two of them which are widely used are being discussed in this paper. Despite their parallels, Microsoft's Visual Basic and Porland International's Delphi as rapid applications development tools, they are clearly oriented toward different audience [17]. Visual Basic is aimed to developers who are using Microsoft products almost exclusively. Delphi provides similar (and in several cases better) functionality for distributed applications, In the same time the two have never looked more alike.

(1) ***Visual Basic 5.0,*** released in March 1997, brought the Visual Basic line even closer to parity with Delphi's performance by way of its long - awaited code compiler. The main characteristics of the Enterprise Edition, are as follows :

*** Description :**

Microsoft's development tool for a distrbuted database applications using Microsoft's products.

*** Strengths :**

- Easy to use, with gentle learning curve.
- Native code compiler debugger for rapid code fixing.
- Includes also, interpretive debugger for rapid code fixing.

*** Weaknesses :**

- Immature distrbuted object implementation for enterprise class applications.
- Database development tools do not do much with other vendors' databases.

(2) ***Delphi 3 Client / server Beta*** : released on May 1997, Porland is doing its own matching, adding ActiveX and distributed applications development capabilities. The main features of this package are :

*** Descrption :**

Porland International's multifier development tool based on object - oriented Pascal.

*** Strengths :**

- Easy to use interface,
- More - mature distributed object technology,
- Object paradigm and repository encourage code form reuse.

*** Weaknesses :**

- Object Pascal has a sleeper learning curve than the Visual Basic.
- Manual migration path for converting applications to ActiveX documents.

The two releases raise the bar for Powersoft's Power Builder 6.0. They also finally offer Windows developers some solid options for rapidly building and deploying distributed applications across their networks and intranets. The improvements in these tools may even be enough to lure developers away from Windows C++ tools.

Also, both tools make it easy to create ActiveX controls and servers, ActiveX documents that can be run from within a Web browser, and executable applications. Both tools can create applications with distrbuted components using Microsoft Distributed Component Object Model (DCOM). Also, both tools support only development of 32-bit Windows 95 and Windows NT applications. 16-bit users of Windows 3.x are left behind to use only older versions of these tools.

In fact, both development tools are very similar when it comes to developing stand - alone a simple client - server applications. The real differences between these tools emerge when they are applied to the task of enterprise wide application development, then the gap between Microsoft's and Porland's strategies for distributed development becomes visible.

4. AUTHORING MULTIMEDIA SOFTWARE :

Multimedia authoring spans a huge range of projects from presentations, computer - based training programs and product demonstrations, to entertainment, and education CD-ROMs, Simulations, Interactive Web sites and Space - alien - destruction games [18].

Therefore, the right authoring environment for these tools is crucial, learning curve, playback and authoring platforms, and project are some of the factors to consider.

There are several mid - to - high end authoring programs, mostly timeline - and icon - based for Windows and Macintosh platforms. All the existed programs combine graphics, sound, animation, video and text into projects with more sophisticated synchroniztion and interactively than is possible with multimedia presentation programs such as Astound and Power - Point. Some such as Macromedia's Director and Advanced Media's Media Master Pro, let the developers to create and edit media right in the applications. Others, such as Apple Media Tool, work only with media created elsewhere. All except Scala Multimedia MM100 let distribute the work with run time player at no charge.

While Director is still the dominant player in the authoring market, most of the applications which were introduced in 1995 are intended to chip away to various niches.

For example, Innovus Multimedia program is geared toward business applications, while less emphasis on fancy graphics and animation and more focus on ease of use database connectivity and network distribution. Media Master Pro, on the other hand, bundles various media editors for an integrating authoring solutions, and Scala Multmedia 100's distinctive multitasking operating system provides the smoothest playback of Windows applications. Finally, looking Glass Software's Media Verse is a good choice for text - oriented applications, and Apple's Media Tool takes ease - of - use honors. and is also one of the best choice for cross - platform work.

5. INTERNET APPLICATIONS SOFTWARE :

The following presentation reviews in a brief manner some aspects of software packages related to ***internet*** and ***intranet*** or **web** applications. The HTML tools and Web middleware, Microsoft's Visual InterDev and language are introduced.

If it is needed to build a Web site, the right software should be allocated to help in the creation of the required site. But just which of the more than 140 software products [19] be selected and used to design the site?

There is a core set of features that it is important to look for in a Web publishing packages. The chosen software package should do all the following :

* Supports the HTML 3-2 standard, and has the flexibility to handle future changes to HTML standards.

* Handles Java applets, ActiveX controls and scripting and image - map creation. It is needed also to look out for the ability to graphically edit forms, tables and frames.

* Has direct support for movies and multimedia files, such as Shakewave, Quick Time and AVI. It is also required to look for image previewing and management features.

* Other useful features to look for are spell checking, syntax coloring (the color coding of HTML tags and attributes), and the ability to edit multiple documents.

* Ideally a software product should have also HTP (and helpfully HTTP) transfer support.

* It should have an uninstall option just in case the developer changes his mind.

It should be noticed that HTML tools range from single text editors to DTD - like programs which give complete graphical control over page design. Deciding which package suits the needed page, the developer will depend on his skills as a programmer and a designer in the meantime. For example, designers will appreciate programs such as NetObjects Fusion because it gives them a DTP - level control over appearance of Web site. On the other hand, programmers will probably prefer packages such as HotMetal Pro which gives them access to the latest HTML and Web site gizmos like ActiveX controls and Java applets.

If someone is working with Web design team, he probably needs more than pick of software. He will need different tools for different members of the team carrying out different tasks.

The current version of HTML is 3-2. The latest update to the standard have essentially been pushed through by Microsoft and Netscape as they constantly revise these browsers.

The version 3-2 of HTML Document Type Defintion (DTD) includes most of these Microsoft and Netscape extension alongside with minor revisions to the structure of some of the new elements to improve cross - platform and cross - browser compatibility. Both Navigator 3.0 and Interment Explorer 3.0 can display documents created with the latest version of HTML.

The information system Web publishing tools or HTML Tools come in four different types which are :

* **HTML editors** : the developers works, here, in text mode and have buttons to add to HTML tags.

* **WYSIWYG editors** : These products work much like word processors, so it is not needed to edit HTML at all.

* **Site Management Tools :** These products focus on the creation and management of Web sites rather than single pages.

* **Web Top Publishing (WTP) :** This category currently has only one package, NetObject Fusion.

In addition to HTML Tools which are very crucial to network applications, there are some developed software packages which are of interest to Internet Web and Intranet applications, such as :

(1) Web middleware :

This technology has evolved to provide help to those who may have a database that needs a Web front - end [20]. These software packages sit between Web server and SQL database, eliminating some of the drudgery of Common Gateway Interface (CGI) programming while improving productivity in applications development.

CGI developed from a need to extend Web beyond simple delivery of HTML documents because enterprises want to be able to process information submitted by Web users in some intelligent way, Also, middeware stems a need for a mechanism to connect a person running a Web browser to programs and files on the server end.

CGI was evolved primarily under UNIX, but was later translated to Windows and Macintosh server environments. This defines how a browser user's input communicate from a Web server process to an external program or script. Using CGI, Webmasters could enrich their sites, connecting users to live data, whether it was from a product database or a weather map.

The problem was that Webmasters had to write separate CGI programs for each task in scripting languages such as C, Visual Basic, and Apple Script. These programs or scripts accomplished similar functions but rarely reuse code. Webmasters needed to live databases, can provide important performance advantages over CGI. The traditional CGI programming uses a new process to be spawned for each invocation of a given script.

For server handling few transactions, this is considered fine, but those handling hundreds of transactions per hour can slow to crawl as memory fills with redundant process. Most Web middleware software settle this problem by lauding a single program, a server plug-in, that remains inconstant communication with the server no matter how many transactions are outstanding. Commonly supported servers include Netscape, through its NSAPL, and Microsoft's Internet Information Server, through ISPAPI.

Therefore, Web middleware products are tools which are considered as rapid application development solutions. Examples of these tools are :

* Porland's Delphi

* Powersoft's Powerbuilder

These two packages use ODBC, the Open Database Connectivity Standard. Other tools exploit the naive network interfaces of database software itself.

(2) Microsoft's Released Visual InterDev (IDE) : [21]

This software package is an integrated Visual development environment for building and Internet applications. This software package brings together a bundle of visual tools to help Web resources. Visual InterDev shares a common development environment with Microsoft's Visual J++ and Visual C++.

This package was designed to meet the following user requirements :

* An integrated tool for higher productivity,

* Built-in database connectivity tool,

* Built-in development, publishing and site management,

* An open architecture based on HTML/HTTP standards,

* Easy back-end integration with client - server system,

* Open database connectivity.

* Extensibility with other tools,

* Scaleable solutions, and

* A team - based development environment.

One of the important aspects of the Visual InterDev is its active server application development. Microsoft uses the term Web application to indicate any Web site that uses active content on server to perform applications such as pulling information of a database, providing user input to database, and dynamically generating HTML in response to user.

Web applications that are built with Visual InterDev typically use the Active Server Pages (ASP) component on server, which contains server - side scripts (either Vbscrits or Jscripts) along with HTML. The ASP pages can perform almost any task that the traditional application can do, e.g. :

* Integrating with database;

* displaying information to users;

* accessing external server components; and

* reading and writing information to files on server.

(3) Java Language :

In this connection, it should be noted that *Java language* has moved from being an unproved but promising technology to a reliable tool for building business applications on Internet and Intranet [22].

Such Microsystems, Jave soft unit uses Jave for corporate developers. Also, Microsoft is working to encapsulate key portions of Windows 95 and NT applications programming interface as Jave class libraries. This would make Windows programs written in C++, Visual Basic or other Windows development environments more easily portable to Java.

CONCLUSION

As noted above, one cannot ignore the key technology trends that are now unfolding in software production. Large software development firms and organizations, especially in advanced countries, are increasing their degree of software automation and improving their efficiency and performance by better software management practices. Those firms which are existed in Egypt for example, should do some extent acquire, purchase, integrate and exploit software development tools to be able to provide their customers with working software prototypes and customers - oriented software. But this technological capalilities require increasing capital, skilled manpower, access to external knowledge, and an organizational maturity, which are not existed in many software development firms and organizations in Egypt up to now.

The following reasons highlight the importance of technological implications on software industry :

(1) The rapid progress of integrated circuits technlolgy decreases hardware cost steadily. Many hardware components that were developed several years ago may be replaced by off - the - shelf software packages utilizing integrated circuits chips. Some components, which do not have off - the - shelf integrated circuits products are preferred to be implemented in the form of Applications Specific Integrated Circuits (ASIC). These measures also greatly increase the hardware reliability.

(2) The wide applications of various types of Central Processing Unit (CPU) chips simplifies the hardware design in the sense that the same hardware architecture can be used to fulfill different kinds of functions with different specialized implementation of software embedded in the CPU chips.

(3) Functions of large enterprise's products are becoming more complex and sophisticated in an open and changing environment. Therefore, they need applications of diversified nature. Intranet and Web software applications are evolved to meet networking requirements.

(4) Various forms of computer management, control and maintenance systems play important roles in operating and maintaining large software based products. They perform complex functions and usually require a considerable amount of effort to be developed and implemented.

BIBLIOGRAPHY

[1] "Software patents", New Scientist, (Feb. 4, 1993) & Microelectronics Monitor. vol., no. 2 (1995), p. 52.

[2] Humphrey, W. S. and Sweet, W. L. "A method for assessing the software engineering capability of contractors" (Pittsburgh, PA: Software Engineering Institute, Carnegie - Mellon University, 1987), pp. 5-6.

[3] ibid., pp. 23-30.

[4] ibid.

[5] Kellner, M. I, and Hansen, G. A. Software process modelling (Pittsburgh, PA; Software Engineering Institute, Carnegie - Mellon University, 1988).

[6] El Hadi, M. M. "Standardization in information technology and telecommunications for open systems interconnection" In : Towards the development of electronic Arabic resources and the challenges of civilization, (Cairo; Academic Bookshop, 1997).

[7] Caswell, D. L. and Grady, R. B. software metrics : establishing a company - wide program, (Englewood - Cliffs, NJ : Prentice - Hall, 1987).

[8] ibid., p. 25.

[9] ibid., p. 199.

[10] ibid., p. 1.

[11] "Software development". Computer Weekly, (Nov. 7, 1991).

[12] ODA. Information Technology R & D: critical trends and issues, (Washington, DC: Office of Technology Assessment, 1985), p. 76.

[13] Business Software Review, (1987), pp. 29-30.

[14] Picardi, A. "CORTEX Application Factory : concepts and facilities", Auebrach Information Management Reference (Boston, MA: Auebrach Publishers, 1987), p. 3.

[15] Young, J. L. "The software foundry : almost too good to be true", Electronics, (Jan. 21, 1988), pp. 1-6.

[16] Padding, Alan, "Application development: benefits of reuse", www. Informationweek. com (March 31, 1997), pp. 1-6.

[17] Feibus, Andy, "Two paths to RAD" www. Informationweek. com. (March 31, 1997), pp. 53-58.

[18] Florio, Chris, "Authoritative authoring software that makes multimedia happens", New Media, vol. 6, No. 12 (Sept. 9, 1996), pp. 67-75.

[19] "The best HTML tool for you" Internet Magazine, (Feb., 1970), pp. 32-36.

[20] Wiggins, Ricard, "Middleware eases Webmaster's burden" New Media, vol. 6, No. 14 (Oct. 28, 1996).

[21] Spencer, Ken, "Application development : one - stop Web creation". www. Informationweek. com. (April 14, 1997), pp. 1-5.

[22] Wilder, Clinton, Patrizie, Andly and Levin, Rich, "Jave power play", www. Informationweek. com. (March 31, 1997), pp. 14-18.